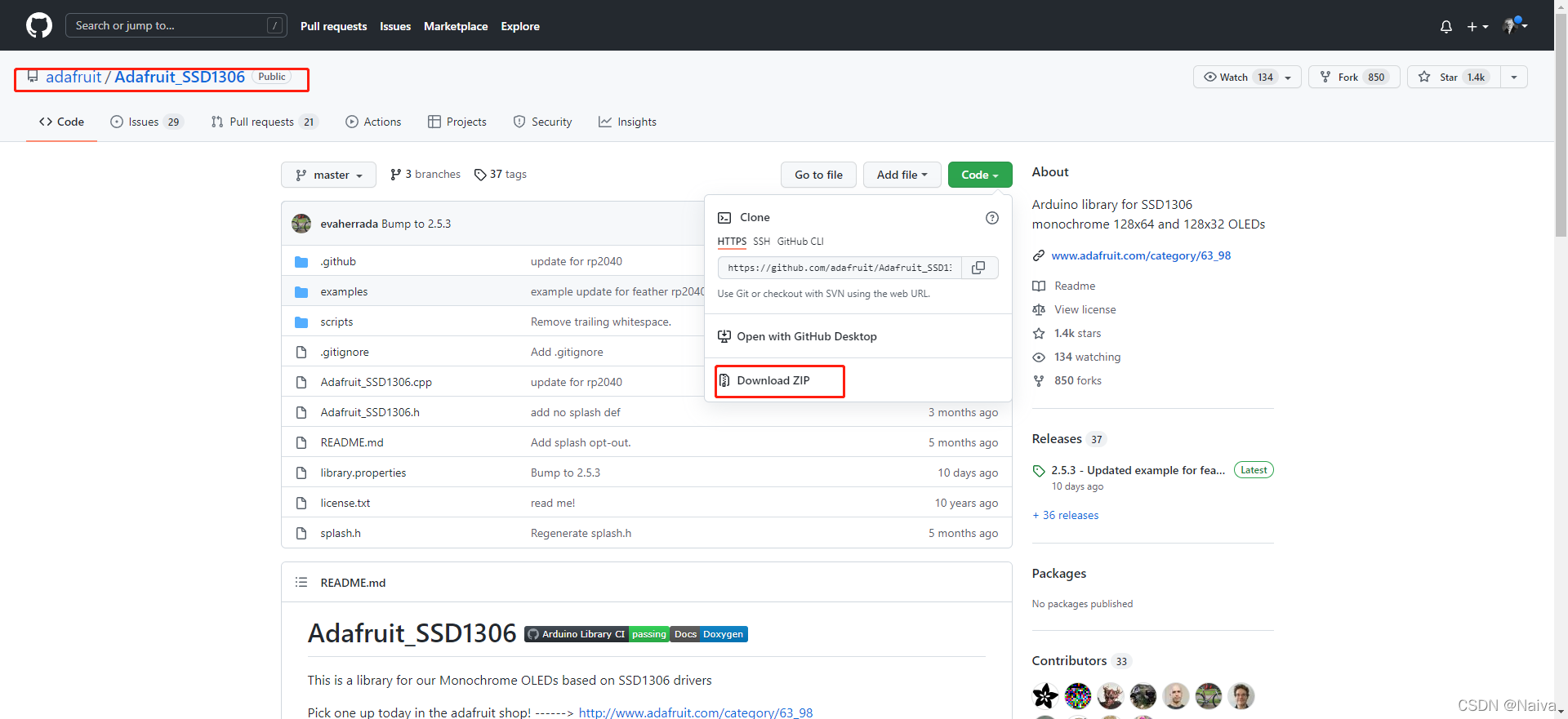
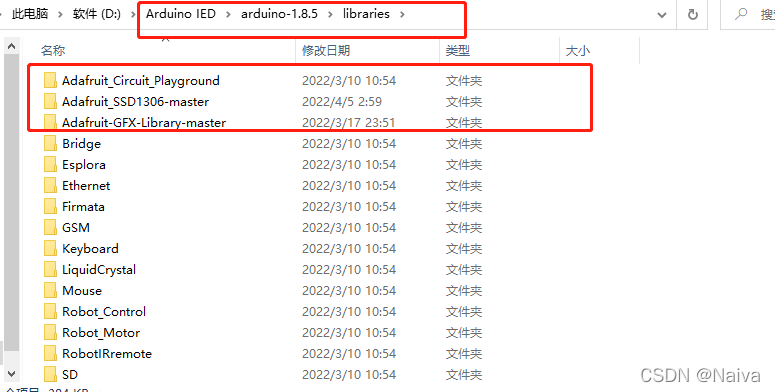
ESP32控制SSD1306OLED显示屏

###### 安装所需库文件

安装 SSD1306 OLED 库 – ESP32(也可以用u8g2库，在Arduino IDE中可以搜索下载该库)。有几个库可用于使用 ESP32 控制 OLED 显示器。我们使用两个 Adafruit 库：[Adafruit\_SSD1306](https://github.com/adafruit/Adafruit_SSD1306) 库和[Adafruit\_GFX](https://github.com/adafruit/Adafruit-GFX-Library) 库。



下载库文件，解压在Arduino IDE安装的对应路径。



SSD1306 是通过I2C协议进行通信的。

什么是I2C协议？

I2C 是用于设备之间通信的双线协议，在物理层面，它由 2 条线组成： SCL 和SDA，分别是时钟线和数据线。也就是说不通设备间通过这两根线就可以进行通信。

背景图案, 表格

描述已自动生成

连线说明（SCL与SDA可选择其他引脚）

图形用户界面, 应用程序

描述已自动生成

电子游戏的截图

描述已自动生成

###### 编译报错：

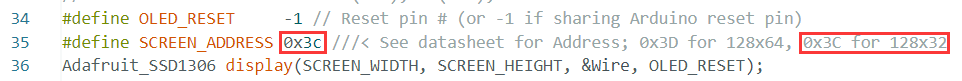
libraries\Adafruit-GFX-Library-master\Adafruit\_GrayOLED.h:30:32: fatal error: Adafruit\_I2CDevice.h: No such file or directory为开发板 ESP32 Dev Module 编译时出错。

**解决方法**：

如果使用较早版本的 Arduino IDE1.8.10 之前，请找到并安装 [Adafruit\_BusIO](https://github.com/adafruit/Adafruit_BusIO)，较新版本会自动处理此先决条件。

###### 记Adafruit\_SSD1306使用自定义i2c引脚

使用Adafruit\_SSD1306，但是网上的关于四脚i2c接口oled屏幕的教程都是千篇一律从官方示例拿来的代码，只有使用硬件i2c引脚的示例，查询源码后发现使用任意gpio引脚驱动的方法，记录如下：



图形用户界面, 文本

描述已自动生成

关键是增加了Wire.begin(SDA,SCL);这一句来使用软件i2c引脚

display.begin中

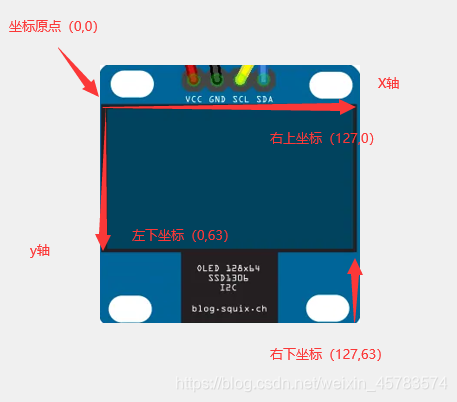
第一个参数不知道什么意思，但不影响使用

第二个参数表示i2c地址，地址见数据表；0x3D用于128x64, 0x3C用于128x32

第三个参数表示是否reset

第四个参数periphBegin比较关键，要为false，否则函数内部将再次调用Wire.begin()

###### OLED显示原理：

OLED其实就是一个M×N的像素[点阵](https://so.csdn.net/so/search?q=%E7%82%B9%E9%98%B5&spm=1001.2101.3001.7020)，需要显示什么就得把具体坐标位置对应的像素点点亮。  


显示图片

图片转二维数组像素点显示

方法：[Arduino应用开发——LCD显示图片\_lcd-image-converter-CSDN博客](https://blog.csdn.net/ShenZhen_zixian/article/details/122682500)

**中文显示**

[esp32 下蓝牙播放音乐歌词的获得\_esp32蓝牙播放音乐-CSDN博客](https://blog.csdn.net/platform/article/details/130845245?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522170443410416800225511202%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=170443410416800225511202&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-1-130845245-null-null.142%5ev99%5econtrol&utm_term=esp32%E8%93%9D%E7%89%99%E6%92%AD%E6%94%BE%E9%9F%B3%E4%B9%90%E8%8E%B7%E5%8F%96%E6%AD%8C%E8%AF%8D&spm=1018.2226.3001.4187)

[ssd1306OLED中文显示-MicroPython-ESP32-利用GB2312字库（非手动取模）\_esp32 加载外部字库-CSDN博客](https://blog.csdn.net/hehedadaq/article/details/117596103?ops_request_misc=&request_id=&biz_id=102&utm_term=esp32%20ssd1306%E6%98%BE%E7%A4%BA%E4%B8%AD%E6%96%87&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-1-117596103.142%5ev99%5econtrol&spm=1018.2226.3001.4187)

**等等还没学：**

[Arduino学习笔记：Adafruit\_SSD1306——OLED学习\_哈哈浩丶的博客-CSDN博客](https://blog.csdn.net/weixin_45783574/article/details/115871082)

[玩转 ESP32 + Arduino (二十五) SSD1306库驱动OLED-CSDN博客](https://blog.csdn.net/finedayforu/article/details/108769900)

[ESP32 入门笔记04： 0.96寸OLED 显示屏 + u8g2库丝滑显示UI (ESP32 for Arduino IDE)\_0.96oled显示屏动态显示\_Naiva的博客-CSDN博客](https://blog.csdn.net/Naiva/article/details/124190441)

**官方文档：**

[ESP32 OLED 显示屏，搭载 Arduino IDE |随机书教程 (randomnerdtutorials.com)](https://randomnerdtutorials.com/esp32-ssd1306-oled-display-arduino-ide/)

###### 附录

**附录A：HelloWorld**

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -1);

void setup() {

  Serial.begin(115200);

  Wire.begin(23,22);

  if(!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C,false,false)) { // Address 0x3D for 128x64

    Serial.println(F("SSD1306 allocation failed"));

    for(;;);

  }

  delay(2000);

  display.clearDisplay();

  display.setTextSize(1);

  display.setTextColor(WHITE);

  display.setCursor(0, 10);

  // Display static text

  display.println("Hello, world!");

  display.display();

}

void loop() {

}

**附录：示例代码**

#include <SPI.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)

// The pins for I2C are defined by the Wire-library.

// On an arduino UNO:       A4(SDA), A5(SCL)

// On an arduino MEGA 2560: 20(SDA), 21(SCL)

// On an arduino LEONARDO:   2(SDA),  3(SCL), ...

#define OLED\_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)

#define SCREEN\_ADDRESS 0x3c ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

#define NUMFLAKES     10 // Number of snowflakes in the animation example

#define LOGO\_HEIGHT   16

#define LOGO\_WIDTH    16

static const unsigned char PROGMEM logo\_bmp[] =

{ 0b00000000, 0b11000000,

  0b00000001, 0b11000000,

  0b00000001, 0b11000000,

  0b00000011, 0b11100000,

  0b11110011, 0b11100000,

  0b11111110, 0b11111000,

  0b01111110, 0b11111111,

  0b00110011, 0b10011111,

  0b00011111, 0b11111100,

  0b00001101, 0b01110000,

  0b00011011, 0b10100000,

  0b00111111, 0b11100000,

  0b00111111, 0b11110000,

  0b01111100, 0b11110000,

  0b01110000, 0b01110000,

  0b00000000, 0b00110000 };

void setup() {

  Serial.begin(9600);

  Wire.begin(/\*SDA\*/23,/\*SCL\*/22);//修改引脚

  // SSD1306\_SWITCHCAPVCC = generate display voltage from 3.3V internally

  if(!display.begin(SSD1306\_SWITCHCAPVCC, SCREEN\_ADDRESS,false,false)) {

    Serial.println(F("SSD1306 allocation failed"));

    for(;;); // Don't proceed, loop forever

  }

  // Show initial display buffer contents on the screen --

  // the library initializes this with an Adafruit splash screen.

  display.display();

  delay(2000); // Pause for 2 seconds

  // Clear the buffer

  display.clearDisplay();

  // Draw a single pixel in white

  display.drawPixel(10, 10, SSD1306\_WHITE);

  // Show the display buffer on the screen. You MUST call display() after

  // drawing commands to make them visible on screen!

  display.display();

  delay(2000);

  // display.display() is NOT necessary after every single drawing command,

  // unless that's what you want...rather, you can batch up a bunch of

  // drawing operations and then update the screen all at once by calling

  // display.display(). These examples demonstrate both approaches...

  testdrawline();      // Draw many lines

  testdrawrect();      // Draw rectangles (outlines)

  testfillrect();      // Draw rectangles (filled)

  testdrawcircle();    // Draw circles (outlines)

  testfillcircle();    // Draw circles (filled)

  testdrawroundrect(); // Draw rounded rectangles (outlines)

  testfillroundrect(); // Draw rounded rectangles (filled)

  testdrawtriangle();  // Draw triangles (outlines)

  testfilltriangle();  // Draw triangles (filled)

  testdrawchar();      // Draw characters of the default font

  testdrawstyles();    // Draw 'stylized' characters

  testscrolltext();    // Draw scrolling text

  testdrawbitmap();    // Draw a small bitmap image

  // Invert and restore display, pausing in-between

  display.invertDisplay(true);

  delay(1000);

  display.invertDisplay(false);

  delay(1000);

  testanimate(logo\_bmp, LOGO\_WIDTH, LOGO\_HEIGHT); // Animate bitmaps

}

void loop() {

}

void testdrawline() {

  int16\_t i;

  display.clearDisplay(); // Clear display buffer

  for(i=0; i<display.width(); i+=4) {

    display.drawLine(0, 0, i, display.height()-1, SSD1306\_WHITE);

    display.display(); // Update screen with each newly-drawn line

    delay(1);

  }

  for(i=0; i<display.height(); i+=4) {

    display.drawLine(0, 0, display.width()-1, i, SSD1306\_WHITE);

    display.display();

    delay(1);

  }

  delay(250);

  display.clearDisplay();

  for(i=0; i<display.width(); i+=4) {

    display.drawLine(0, display.height()-1, i, 0, SSD1306\_WHITE);

    display.display();

    delay(1);

  }

  for(i=display.height()-1; i>=0; i-=4) {

    display.drawLine(0, display.height()-1, display.width()-1, i, SSD1306\_WHITE);

    display.display();

    delay(1);

  }

  delay(250);

  display.clearDisplay();

  for(i=display.width()-1; i>=0; i-=4) {

    display.drawLine(display.width()-1, display.height()-1, i, 0, SSD1306\_WHITE);

    display.display();

    delay(1);

  }

  for(i=display.height()-1; i>=0; i-=4) {

    display.drawLine(display.width()-1, display.height()-1, 0, i, SSD1306\_WHITE);

    display.display();

    delay(1);

  }

  delay(250);

  display.clearDisplay();

  for(i=0; i<display.height(); i+=4) {

    display.drawLine(display.width()-1, 0, 0, i, SSD1306\_WHITE);

    display.display();

    delay(1);

  }

  for(i=0; i<display.width(); i+=4) {

    display.drawLine(display.width()-1, 0, i, display.height()-1, SSD1306\_WHITE);

    display.display();

    delay(1);

  }

  delay(2000); // Pause for 2 seconds

}

void testdrawrect(void) {

  display.clearDisplay();

  for(int16\_t i=0; i<display.height()/2; i+=2) {

    display.drawRect(i, i, display.width()-2\*i, display.height()-2\*i, SSD1306\_WHITE);

    display.display(); // Update screen with each newly-drawn rectangle

    delay(1);

  }

  delay(2000);

}

void testfillrect(void) {

  display.clearDisplay();

  for(int16\_t i=0; i<display.height()/2; i+=3) {

    // The INVERSE color is used so rectangles alternate white/black

    display.fillRect(i, i, display.width()-i\*2, display.height()-i\*2, SSD1306\_INVERSE);

    display.display(); // Update screen with each newly-drawn rectangle

    delay(1);

  }

  delay(2000);

}

void testdrawcircle(void) {

  display.clearDisplay();

  for(int16\_t i=0; i<max(display.width(),display.height())/2; i+=2) {

    display.drawCircle(display.width()/2, display.height()/2, i, SSD1306\_WHITE);

    display.display();

    delay(1);

  }

  delay(2000);

}

void testfillcircle(void) {

  display.clearDisplay();

  for(int16\_t i=max(display.width(),display.height())/2; i>0; i-=3) {

    // The INVERSE color is used so circles alternate white/black

    display.fillCircle(display.width() / 2, display.height() / 2, i, SSD1306\_INVERSE);

    display.display(); // Update screen with each newly-drawn circle

    delay(1);

  }

  delay(2000);

}

void testdrawroundrect(void) {

  display.clearDisplay();

  for(int16\_t i=0; i<display.height()/2-2; i+=2) {

    display.drawRoundRect(i, i, display.width()-2\*i, display.height()-2\*i,

      display.height()/4, SSD1306\_WHITE);

    display.display();

    delay(1);

  }

  delay(2000);

}

void testfillroundrect(void) {

  display.clearDisplay();

  for(int16\_t i=0; i<display.height()/2-2; i+=2) {

    // The INVERSE color is used so round-rects alternate white/black

    display.fillRoundRect(i, i, display.width()-2\*i, display.height()-2\*i,

      display.height()/4, SSD1306\_INVERSE);

    display.display();

    delay(1);

  }

  delay(2000);

}

void testdrawtriangle(void) {

  display.clearDisplay();

  for(int16\_t i=0; i<max(display.width(),display.height())/2; i+=5) {

    display.drawTriangle(

      display.width()/2  , display.height()/2-i,

      display.width()/2-i, display.height()/2+i,

      display.width()/2+i, display.height()/2+i, SSD1306\_WHITE);

    display.display();

    delay(1);

  }

  delay(2000);

}

void testfilltriangle(void) {

  display.clearDisplay();

  for(int16\_t i=max(display.width(),display.height())/2; i>0; i-=5) {

    // The INVERSE color is used so triangles alternate white/black

    display.fillTriangle(

      display.width()/2  , display.height()/2-i,

      display.width()/2-i, display.height()/2+i,

      display.width()/2+i, display.height()/2+i, SSD1306\_INVERSE);

    display.display();

    delay(1);

  }

  delay(2000);

}

void testdrawchar(void) {

  display.clearDisplay();

  display.setTextSize(1);      // Normal 1:1 pixel scale

  display.setTextColor(SSD1306\_WHITE); // Draw white text

  display.setCursor(0, 0);     // Start at top-left corner

  display.cp437(true);         // Use full 256 char 'Code Page 437' font

  // Not all the characters will fit on the display. This is normal.

  // Library will draw what it can and the rest will be clipped.

  for(int16\_t i=0; i<256; i++) {

    if(i == '\n') display.write(' ');

    else          display.write(i);

  }

  display.display();

  delay(2000);

}

void testdrawstyles(void) {

  display.clearDisplay();

  display.setTextSize(1);             // Normal 1:1 pixel scale

  display.setTextColor(SSD1306\_WHITE);        // Draw white text

  display.setCursor(0,0);             // Start at top-left corner

  display.println(F("Hello, world!"));

  display.setTextColor(SSD1306\_BLACK, SSD1306\_WHITE); // Draw 'inverse' text

  display.println(3.141592);

  display.setTextSize(2);             // Draw 2X-scale text

  display.setTextColor(SSD1306\_WHITE);

  display.print(F("0x")); display.println(0xDEADBEEF, HEX);

  display.display();

  delay(2000);

}

void testscrolltext(void) {

  display.clearDisplay();

  display.setTextSize(2); // Draw 2X-scale text

  display.setTextColor(SSD1306\_WHITE);

  display.setCursor(10, 0);

  display.println(F("scroll"));

  display.display();      // Show initial text

  delay(100);

  // Scroll in various directions, pausing in-between:

  display.startscrollright(0x00, 0x0F);

  delay(2000);

  display.stopscroll();

  delay(1000);

  display.startscrollleft(0x00, 0x0F);

  delay(2000);

  display.stopscroll();

  delay(1000);

  display.startscrolldiagright(0x00, 0x07);

  delay(2000);

  display.startscrolldiagleft(0x00, 0x07);

  delay(2000);

  display.stopscroll();

  delay(1000);

}

void testdrawbitmap(void) {

  display.clearDisplay();

  display.drawBitmap(

    (display.width()  - LOGO\_WIDTH ) / 2,

    (display.height() - LOGO\_HEIGHT) / 2,

    logo\_bmp, LOGO\_WIDTH, LOGO\_HEIGHT, 1);

  display.display();

  delay(1000);

}

#define XPOS   0 // Indexes into the 'icons' array in function below

#define YPOS   1

#define DELTAY 2

void testanimate(const uint8\_t \*bitmap, uint8\_t w, uint8\_t h) {

  int8\_t f, icons[NUMFLAKES][3];

  // Initialize 'snowflake' positions

  for(f=0; f< NUMFLAKES; f++) {

    icons[f][XPOS]   = random(1 - LOGO\_WIDTH, display.width());

    icons[f][YPOS]   = -LOGO\_HEIGHT;

    icons[f][DELTAY] = random(1, 6);

    Serial.print(F("x: "));

    Serial.print(icons[f][XPOS], DEC);

    Serial.print(F(" y: "));

    Serial.print(icons[f][YPOS], DEC);

    Serial.print(F(" dy: "));

    Serial.println(icons[f][DELTAY], DEC);

  }

  for(;;) { // Loop forever...

    display.clearDisplay(); // Clear the display buffer

    // Draw each snowflake:

    for(f=0; f< NUMFLAKES; f++) {

      display.drawBitmap(icons[f][XPOS], icons[f][YPOS], bitmap, w, h, SSD1306\_WHITE);

    }

    display.display(); // Show the display buffer on the screen

    delay(200);        // Pause for 1/10 second

    // Then update coordinates of each flake...

    for(f=0; f< NUMFLAKES; f++) {

      icons[f][YPOS] += icons[f][DELTAY];

      // If snowflake is off the bottom of the screen...

      if (icons[f][YPOS] >= display.height()) {

        // Reinitialize to a random position, just off the top

        icons[f][XPOS]   = random(1 - LOGO\_WIDTH, display.width());

        icons[f][YPOS]   = -LOGO\_HEIGHT;

        icons[f][DELTAY] = random(1, 6);

      }

    }

  }

}